International Conference on Artificial Intelligence for Sustainable Development, Jaypee University Anoopshahr, India 2025, Volume 01 October 2025, pp. 21-33

TEST CASE SELECTION TECHNIQUES FOR SOFTWARE PRODUCT LINE USING AI TECHNIQUES: A SURVEY

Amit Kumar [1], Pratap Singh [2], Satendra Kumar [3]

[1, 2] *Department of CSE, Quantum University Roorkee, Uttarakhand, India*

[3] *Department of CSE, Meerut Institute of Engineering & Technology, Meerut, India*

amitcse86@gmail.com(https://orcid.org/0009-0003-1564-4837),
partap.cse@quantumeducation.in((https://orcid.org/0009-0007-8141-2355),
satendra04cs41@gmail.com(https://orcid.org/0000-0003-2336-8839)

Abstract. Software Product Line Engineering (SPLE) is an approach in the field of software engineering where industrial production techniques are applied, and software development is gradually altered. Reuse and maintaining the traceability of variant features of all products belonging to the same product line family are fundamental requirements. The ideal approach to software development is to utilize software engineering techniques to precisely and successfully regulate it. In this research, we will describe previous work of different researchers related to test case selection and software product lines, focusing more on the TCS technique, its benefits, and its challenges.

Keywords: software product line; techniques of test case selection; feature model; challenges of TCS

INTRODUCTION

Developing software products with similar features is simplified by the Software Product Line (SPL) model of software engineering. The main goal of developing a system as a software product line is to maximize software reusability to create a software product that can be customized to meet various needs [22].

The necessity to create several related software products, rather than a single product, is a growing trend in software development. This is due to several factors. First, to accommodate different legal or cultural contexts, as well as linguistic differences, the user interfaces of products designed for the global market must be adjusted. Second, software reuse must be enhanced because software developers cannot create a new product from scratch for every new customer, given time and expense restrictions.

Software Product Line Testing (SPLT) frequently focuses solely on the selection of test cases that address the features or interactions of the product. Nonetheless, choosing test cases with enhanced fault-revealing capabilities can result in more efficient testing. Finding a test suite that maximizes the achievement of a particular aim, such as less test time or expense, while maintaining complete feature coverage, thus becomes a key goal in SPLT.

To select test cases for software product lines, the following techniques can be applied:

1. Feature-Based Test Case Selection
2. Combinatorial Testing
3. Variability-Based Test Case Selection

4. Test Case Prioritization
5. Reuse of Test Cases across Configurations
6. Model-Based Testing
7. Change Impact Analysis

The process of developing a Software Product Line (SPL) is depicted in figure1. SPL Development uses feature binding and configuration to create a Product Line. Then, using runtime binding, products are produced with certain active, new, or deactivated features. Reduced test suites are used to guarantee effective testing of goods with bound features that are customized.

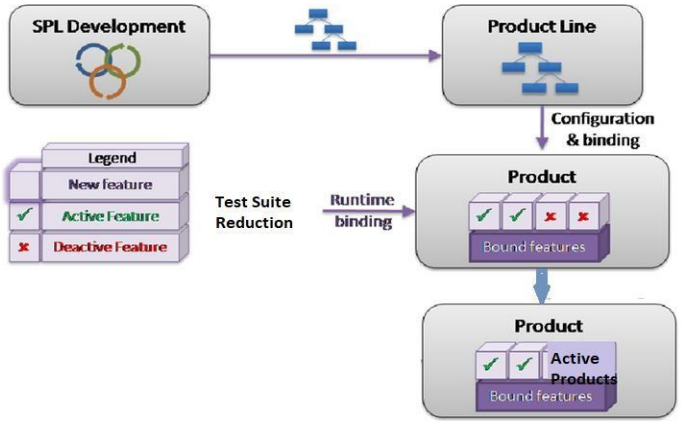

Figure 1. Product Software Line

PRELIMINARY WORK

Feature Modeling

A feature is a system attribute that is important to a particular stakeholder. A feature may be a technical function or function group, a non-functional (quality) attribute, or a necessity, depending on the interests of the stakeholders. SPL generates a number of related products using a feature model (FM). Each product has a special combination of features.

Four feature groups are used to derive the SPL products, and they are:

- Mandatory feature group
- Optional feature group
- OR feature group
- XOR (Alternative) feature group

The following guidelines are used to choose the characteristics.

In the event that the parent feature is chosen:

- Every required sub-feature needs to be chosen.
- Sub-features that are optional can be chosen or not.
- From the alternative feature group, only one feature ought to be chosen.
- The OR feature group should have at least one feature chosen.

A feature model diagram of a smartphone's setup possibilities is shown in figure 2.Functions including calls, multimedia, screen, and GPS are displayed, along with optional and required functions. Basic vs. high_resolution under screen, for example, or OR (camera/photo/music under multimedia) are used to categorize certain functions.

The bottom constraints specify cross-tree dependencies: basic ⇒ GPS and camera ⇒ high_resolution.

Lastly, one of the biggest challenges is testing awareness simultaneously. The majority of product lines offer many more possible variations than are currently in use, and it is occasionally impossible or a waste of time to test every single variation [19]. Where there are a large variety of variants, testing just the ones that were created is already a difficult drawback.

Coordination of testing and variant manufacture is still required, though. One strategy is to cooperatively develop quality variants for the (software) response house variants, thereby creating a parallel solution house that is based on the feature model. However, for a number of product lines, lowering the amount of effort required to look at things remains a somewhat unresolved issue.

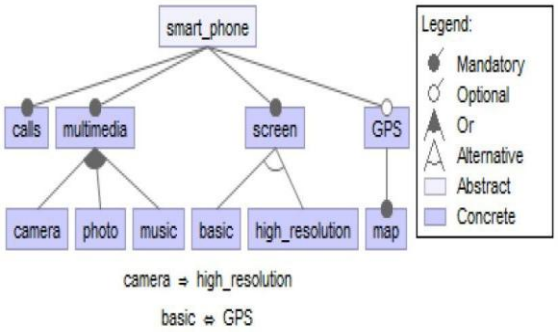

Figure 2. Feature Model of Smartphone SPL

SPL Testing Methodology

Conventional testing methods struggle to scale effectively for Software Product Lines (SPLs) because of the inherent variability and interactions among features. To guarantee reliability while preserving efficiency, a structured approach to testing SPLs is essential. Current techniques for testing SPLs encompass combinatorial testing, model-based testing, and sampling methods like pairwise and t-wise testing. Nevertheless, there are still shortcomings in optimizing the selection of test cases to achieve a balance between coverage and execution costs.

A maximum of 2n products of an SPL can be produced by an FM with n characteristics. It is not possible to test every product as the size of software products increases.

There are the following testing methodologies applied to SPL.

- Test Case Selection (TCS) techniques—"Screen"
- Test Case Prioritization (TCP) techniques—“Order”
- Test Suite Reduction (TSR) or Test Case Reduction (TCR) techniques—“Remove”
- Test all.

Techniques for Test Case Selection (TCS) are crucial for enhancing testing efficiency in the development of Software Product Lines (SPL), where various products are built upon a shared core with distinct variations. These techniques not only aid in providing comprehensive coverage of various configurations but also help minimize redundancy.

In the domain of Software Product Line (SPL), Test Case Prioritization (TCP) is a method employed to arrange test cases to optimize both the effectiveness and efficiency of testing processes [20-21]. Given that SPL encompasses a collection of interconnected software products with differing functionalities, TCP plays a crucial role in pinpointing the most critical test cases initially, thereby accelerating fault detection and minimizing the duration of testing.

In Software Product Lines (SPLs), Test Suite Reduction (TSR) is a method employed to decrease the number of test cases while still ensuring a high level of fault detection. Testing every possible configuration is impractical since SPLs consist of various software product variants that share certain components and differ in others. By eliminating unnecessary test cases, TSR enhances testing efficiency while still adequately testing essential functionalities.

It is very difficult to test all cases in the product line due to increased time in the development of software and wastage of human effort. So researchers work on these techniques to accelerate the software development with minimum human effort and reliable software.

A. Test Case Selection Technique

The process of test case selection entails identifying a portion of test cases from a pre-existing test suite to run on a particular variant of a Software Product Line (SPL). The objective is to enhance fault detection while reducing testing expenses by eliminating redundant or superfluous tests.

The importance of the TCS technique is

Scalability: Because of the combinatorial explosion, it is impractical to evaluate every variant of a product.

Efficiency: It minimizes the time and resources required for testing, leading to cost savings while maintaining effective fault detection.

Regression testing: This approach identifies appropriate test cases for modified features.

Now we are discussing the challenge of TCS.

Feature Combinations: With an increase in the number of features, the variety of product variations grows exponentially, leading to a dramatic rise in the possible test cases. This creates challenges in selecting a suitable set of test cases that effectively cover all interactions between the features.

Time and Resource Constraints: Testing every possible product configuration might not be feasible due to resource constraints like time and processing power. As a result, methods that can reduce the test set's size while maintaining thorough coverage must be used.

Redundancy Minimization: Inefficient use of resources may result from test cases that overlap because they may assess the same feature combinations. The goal of good test selection techniques is to reduce redundancy while maintaining the efficacy of the testing procedure.

Recent studies have investigated various strategies to tackle these challenges, including:

Machine Learning for Test Case Selection: By using machine learning algorithms, test suite selection can be improved by determining which test cases are most likely to reveal defects.

Integration of Continuous Testing: As DevOps and continuous integration techniques gain traction, automated and continuous testing play a crucial role in overseeing the selection of test cases for Software Product Lines (SPL). The importance of methods that can adjust to frequent changes in features or configurations is growing.

Reducing Test Explosion: To prevent an excessive rise in test cases within large SPLs, techniques like combinatorial optimization, which generate a minimal set of test cases that cover all potential combinations, are being improved.

Software product line engineering

Nowadays, people want to accomplish more with less work, which has increased demand for software product lines (SPL) in the software sector. Instead of concentrating only on individual software products, there is an increasing need to develop similar multiple software products. A number of factors are responsible for this trend. Customized user interfaces are required when products are meant for an international market because they must be made to accommodate different languages, legal systems, and cultural contexts. Furthermore, software reusability has grown in significance as developers frequently encounter financial and temporal limitations that make it impractical to develop completely new products from the ground up for every single client.

Among the many benefits of Software Product Line Engineering (SPLE) are increased productivity, reduced expenses, and better product quality.

The basis of the SPLE is the modelling of commonalities and variations between variants of software pro ducts. FMS [16, 17] is often used for this. SPL also includes the design and management of variable software architectures and their components.

RELATED WORK

In 2023, Shahbaa and Raghda do not particularly address test case selection approaches for software product lines; they concentrate on choosing the best test cases using artificial intelligence techniques, specifically using a genetic algorithm for prioritization and a modified crow search algorithm.

Table 1. Recent Research

S. No.	Publi cation Year	Refere nces N0.	Findings	Research Gap	Objectives

1	2011	[1]	Proposed a goal-oriented test case selection approach. Prioritized features based on stakeholder importance for testing.	Empirical evaluation of the proposed approach is ongoing. The correlation between error coverage and test case prioritization needs investigation.	Reduce required test cases for product line testing. Maintain acceptable error detection coverage during testing.
2	2013	[2]	Methodology reduces test case selection effort significantly. Test engineers positively view the methodology's adoption.	Future evaluation in other product lines is needed. A thorough effectiveness analysis against manual processes is required.	Automate test case selection for product lines. Reduce selection effort while maintaining testing coverage.
3	2013	[3]	Automated methodology reduces test selection time and increases effectiveness. Test engineers positively view the methodology's adaptation in practice.	Innovative application of FM and CFM in literature is unexplored. The need for practical value in questionnaire design was identified.	Propose a systematic methodology for test case selection.
4	2014	[4]	The results demonstrated that the approach significantly reduced test selection time and achieved higher effectiveness in terms of feature coverage, feature pairwise coverage, and fault detection compared to the manual process.	Innovative application of FM and CFM in literature unexplored. The need for practical value in questionnaire design was identified.	Propose a systematic methodology for test case selection. Evaluate the effectiveness and cost of automated selection versus manual process.
5	2015	[5]	The research examines the difficulties associated with selecting automated test cases in product lines, emphasizing the systematic capture of variability and the identification of reusable test cases for newly developed products.	An incomplete feature model requires more detailed information. Cannot select test cases bound to early requirements.	Automate test case selection for product lines. Reduce selection effort while maintaining testing coverage.
6	2017	[6]	A mutation-based approach reveals faults not found in pairwise testing. Tool FMTS supports feature mutation testing and evaluation.	Higher-order mutation testing needs further exploration. Operators for absent features are not adequately addressed.	Present a deeper description of the mutation-based approach. Provide complete experimental results for evaluating the approach.
7	2017	[7]	Improved efficiency, reduced execution time, and good fault detection with historical data and ML integration.	Lack of ML use in real test pipelines, insufficient model comparisons, limited empirical results, and static models.	To enhance regression test selection using a logistic regression model that predicts test case importance.

8	2019	[8]	Three test case selection methods are proposed for time-budget scenarios. Coverage information improves fault detection under low time budgets.	The identified research gap is the lack of efficient techniques for choosing a few test cases from a large Software Product Line (SPL) test suite while maintaining sufficient structural code coverage across product line variations, especially when time constraints are tight.	Propose test case selection methods for product lines. Analyze methods using structural coverage information efficiently.
9	2019	[9]	Collaborative filtering improves test case prioritization and reduction. Ontology aids in storing product configurations effectively.	The problem of effectively testing software product lines (SPLs), which can produce an exponentially high number of product configurations because of different feature combinations, is discussed by the author.	Reduce the number of products for testing efficiency. Prioritize test cases for effective fault detection.
10	2020	[10]	The automated method reduces retest cases by 22.4% on average. No fault-revealing test cases were missed during selection.	Scalability to Large Product Lines and Handling Dynamic Variability	Propose an automated source code-based regression test selection method. Improve efficiency of SPL regression testing by reducing test cases.
11	2022	[11]	The improved algorithm reduces redundant test executions effectively. Method applicable for initial and subsequent product family testing. Proposed methods improve test case selection efficiency and quality.	Extend method applicability to finer granularity levels.	Reduce redundant test executions in software product lines. Propose a practical process for SPL testing.
12	2023	[12]	Proposed methods improve test case selection efficiency and quality. Selected test cases reduce testing duration significantly.	The important problem of choosing test cases optimally to cut down on software testing time and effort without sacrificing quality is covered in the research study.	Minimize time and effort in software testing. Select test cases to uncover most bugs efficiently.
13	2023	[13]	The tool uses AI techniques to choose the best test cases. The selection of test cases is improved by an improved crow search algorithm.	Hybridization of boundary value and branch coverage for fitness function. Utilization of the crow's awareness probability in test case selection.	Build a tool for optimal test case selection. Use AI techniques for efficient error detection.
14	2024	[14]	This study looks at the latest developments in fuzzing research. The main study covers the optimization of several fuzzing phases in software testing as well as the test case validation workflow.	Challenges such as the integration of AI into existing testing frameworks, the need for standardized AI models, and the requirement for large, diverse datasets for training AI systems are noted.	The Study identifies the primary objective as enhancing the efficiency and effectiveness of software testing processes through

					AI-driven methodologies.
15	2025	[[15]	The study examines several test suite reduction techniques for software product lines, emphasizing how well they work to reduce test cases while maintaining fault detection.	The study highlights research gaps in the restricted handling of complicated feature interactions and the scalability of reduction techniques for big and complex software product lines.	The purpose of the study is to carefully investigate and categorize current test suite reduction techniques for software product line testing.

A novel tool was developed that utilizes artificial intelligence techniques to select optimal test cases. This tool aims to increase the efficacy and efficiency of the software testing process, addressing the need for scalable and applicable testing methods. The research introduced the crow search algorithm for selecting test cases. This algorithm was modified and improved to better generate and select test cases that cover the basic paths of the program. The improvements were based on a hybrid approach that combines boundary value criteria and branch coverage in the fitness function calculation [13].

In 2024, Mani and Padmanabhan focus on fuzzing approaches powered by artificial intelligence for software test case optimization, with a particular emphasis on data organization, test case development, and validation. However, it doesn't particularly include test case selection methods for software product lines using AI [14].

In 2025, Satendra Kumar and Viplesh examine a range of Test Suite Reduction (TSR) methods within Software Product Lines (SPLs), with the objective of minimizing the number of test cases/products required for testing while maintaining coverage and fault detection. The study classifies techniques according to their utilization of feature models, constraints, and redundancy among variants; it also addresses combinatorial, optimization, and constraint-based strategies. Furthermore, the authors emphasize the trade-offs between reduction and the risk of overlooking feature interactions, cost and automation, and identify scalability and tool support as significant challenges [15].

The study by Mahmudul Islam and Farhan Khan looks at how AI, particularly machine learning and deep learning, can be used to automate a range of software testing jobs, enhance performance, and expedite testing processes [16].

Instead of specifically discussing test case selection strategies for software product lines using AI, Kiyoshi Ueda, Riku Shikama, and Yuki Shimizu concentrate on automatic test case generation from requirement specifications using machine learning, stressing the selection of structured training data based on cosine similarity. The paper's main strategy is an automated system that chooses regression tests according to source code modifications. In order to optimize the testing process, this approach focuses on determining which test cases are pertinent to the code changes [17].

In order to minimize test space and improve error coverage by giving priority to crucial features and configurations for testing, Alireza Ensan et al. provide a goal-oriented method for choosing and ranking test cases in software product lines. The idea of Software Product Line (SPL) engineering, a technique for developing a group of software products with similar features, is covered in the article. This method reduces development time and expenses by reusing code. He used a different approach and reduced the number of test cases [1].

By employing Feature Model for Testing (FMT) and Component Family Model for Testing (CFMT), Shuai Wang et al. provide a methodical approach to automated test case selection that guarantees thorough coverage of testing functionalities in product lines while drastically cutting down on selection time and effort. The findings demonstrated that the methodology lowers reliance on the subject expertise of individual test engineers while also reducing the selection effort. This is important because it keeps the selection of test cases consistent even when team members shift [2]

By creating hash-based traceability links between test cases and source code, Pilsu Jung et al.'s ActSPL automates the selection of reusable test cases for software product lines, achieving 100% precision and 62% recall while drastically cutting down on testing time [10].

As I reviewed the literature, I discovered that testing in a Software Product Line (SPL) may present greater challenges compared to testing individual systems. Conducting tests on each individual SPL product would be beneficial; however, the associated costs make it impractical to implement. As the number of options increases, the potential variety of products will also grow exponentially, typically based on a feature model. Due to time and cost constraints, the range of alternatives is producing hundreds of different goods. Various methods are planned to lower the cost of testing. Test case reduction (TCR) approaches are among them.

- Study and analysis of test case selection techniques for SPLT.
- Screen the test cases/products of an SPL using AI techniques
- Reduce the efforts of testers to test the products of SPLs.
- Reduce the test space or number of products for SPLs.
- Study different SPL tools (SPLOT, Feature IDE, FAMA-FW etc.). Implement, validate, and analyze our approaches to these tools.
- Improve the quality of products of SPLs so that bug-free products can be achieved.

SIGNIFICANCE OF SURVEY

The aim of the literature survey is to identify, evaluate, and understand all the current research that is relevant to a specific topic, questions, or any other area of interest (SPL testing). In this research, we review all the efforts made by software engineering researchers for SPL testing and follow the process and criteria for gathering primary data (including technical reports, case studies, etc.).

Rather than constructing each software product from scratch, a Software Product Line (SPL) is a set of related software products that are created by reusing a common set of features. One of the important principles of Software Product Line Engineering (SPLE) is reusability [23]. The unique features of each product in a Software Product Line (SPL) describe it. A feature describes a software product's behavior and capabilities. To represent SPLs, feature models (FMs) are frequently used. These models describe the features and how they relate to one another, thereby encapsulating information about every product in an SPL. Information from FMs is retrieved with computer assistance as part of automated FM analysis. Among other things, this analysis makes it easier to examine SPL properties like complexity, variability, and consistency. Several methods, tools, and techniques for FM analysis have been developed over the last 20 years. [7].

This paper introduces TCP methodologies for software product lines (SPLs). Our research is driven by several perspectives.

- What methods have been suggested for the selection of test cases in Software Product Lines (SPLs)?
- What advancements have been made in the research of test case selection techniques?
- What strategies can we implement to enhance the efficiency and effectiveness of testing for SPLs?

RESEARCH METHODOLOGY

This survey was carried out using test case selection (TCS) techniques tailored for software product lines (SPL). Even though a lot of research papers have been written, none of them have specifically addressed TCS methods in relation to SPL. We therefore hope that this work will pique researchers' interest. Even though the methods, benefits, logistics, and constraints of SPL have been covered in a number of studies, our discussion will only focus on the most important elements of TCS techniques that apply to SPL.

Data Sources and Search Tactic

Using the following methodology, we created a search string to find relevant materials for our work. Develop the key terms associated with our subjects. In place of the main term, use related terms.Utilize alternative spellings to locate pertinent information by employing the Boolean operator "OR." Main terms

are connected through the Boolean operator "AND". Various combinations of related terms were utilized for the search terms. The search strings developed for this strategy are outlined below. We created search strings after compiling the resources relevant to our investigation. Better search features are made possible by the dynamic nature of digital literature sources. In order to obtain pertinent materials, the main databases used in this process were IEEExplore, Springer Link, Science Direct, and ACM Digital Library. Not every digital source could be found using a single search term. As a result, we used unique search strings for each database (IEEE, Springer, Elsevier, and ACM publications), as mentioned by others [22, 17]. Two researchers worked together to complete this task, and they had in-depth conversations at every turn to improve the final search strings until we were completely happy with how effective they were.

We conducted a search of key publications from the following digital databases: 1. Google Scholar; 2. Web of Science; 3. Research Gate; 4. Springer Link; 5. Science Direct; 6. EI Compendex/Inspec; 7. IEEExplore; 8. Google (Cite Seer Library); and 9. ACM Digital.

Search strings were utilized within the search engines mentioned in a previous section to identify relevant data. The selection of studies was conducted through a screening process that involved filters to identify pertinent studies using the specified keywords. This process also included refining the search results by eliminating titles of studies, as illustrated in Fig. 3. Subsequently, studies were excluded after a review of their abstract and conclusion sections. The studies included in the final selection met the following criteria:

- Studies were excluded based on the following criteria:
- SPL methods lacking adequate information regarding testing exertion.
- Duplicate studies published across multiple papers, with only the most recent version retained.
- Studies that do not provide comprehensive details on concepts and exercises related to SPL testing.
- Additionally, studies that have already been included from other sources were also excluded.

Study Material Selection

In our search, we gathered data indicating that significant research efforts have been made over the past decade. As illustrated in table 2, a considerable number of conferences and workshops have been organized to support researchers in the field of Software Product Lines (SPL). Numerous organizations have facilitated these events to enhance research, allowing participants to identify challenges and explore solutions proposed by various researchers [22]. Notably, the primary sources for these workshops and conferences are SPLT (Software Product Line Testing) and SPLC (Software Product Line Conference), from which we obtained over 50 papers covering diverse topics such as testing, selection, and configuration within Software Product Lines.

In this discussion, our review of various papers revealed that the methods and techniques suggested by numerous authors primarily address theoretical issues, with no practical problems being highlighted. However, over the past decade, industries have significantly contributed to this field, providing researchers with a genuine platform to confront real-world challenges. This has enabled them to develop solutions through various methods and techniques, as evidenced by several industrial case studies conducted by different authors.

RESULT AND DISCUSSION

According to Table 2 data, conference publications—of which there were 185 articles produced between 2010 and 2025—are the most common way that research is shared. This suggests that presenting work at conferences is strongly preferred, perhaps because of their increased visibility and speed of turnaround. Despite a considerable uptick in 2024 with 19 articles, symposium publications have been comparatively stable over the years, indicating a noteworthy event or heightened focus on symposiums in that year. With peaks in 2013, 2021, and 2023 (7 each), workshop publications exhibit a cyclical pattern. However, they are completely absent in years such as 2010, 2015, and 2020, suggesting irregular participation or shifting relevance over time.

Table 2. Research work published in a different year

Publications Year	Conference	Symposium	Workshop	Journal	Industrial Case Study

2010	6	2	--	5	1
2011	9	2	3	1	--
2012	20	2	4	--	--
2013	15	3	6	1	--
2014	15	1	2	3	2
2015	13	3	--	--	--
2016	17	2	4	1	1
2017	4	1	4	2	--
2018	13	2	3	1	1
2019	8	3	2	2	--
2020	4	3	--	3	1
2021	17	4	7	2	--
2022	10	3	4	2	--
2023	14	2	7	2	--
2024	17	19	3	3	2
2025	3	2	1	1	--
Total	185	54	50	29	8

The number of journal publications, though limited (29 in total), exhibits occasional peaks in 2010 and 2020, indicating a consistent but slower level of engagement with peer-reviewed journals. Industrial case studies are the least frequently published, with only 8 outputs over a span of 16 years, which implies a lack of collaboration between industry and academia or a reduced emphasis on applied research. The year 2024 is notable for being the most productive, showcasing high output across all categories, particularly in symposiums. Conversely, 2025 experiences a marked decrease in all categories, possibly due to it being a partial or ongoing year. Overall, the findings indicate a strong focus on rapid academic dissemination, while highlighting potential opportunities to enhance journal publications and industry collaboration.

Figure 3 illustrates the number of publications across different sources. Conference publications are the highest, approaching 190. Symposium and workshop publications are nearly equal, around 50 each. Journal and industrial case study publications are the lowest, with industrial case studies contributing the least. Figure 4 shows the number of research papers published from 2010 to 2025 across five categories: Conference, Symposium, Workshop, Journal, and Industrial Case Study. Conference papers consistently have the highest count each year. Symposium publications spike sharply in 2025, surpassing all other categories. Other sources remain relatively stable with lower publication counts throughout the years.

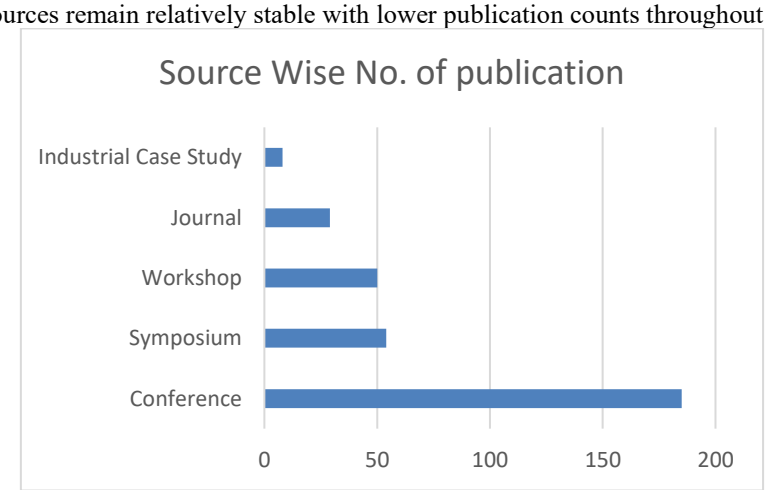

Figure 3. Research work collected from different sources

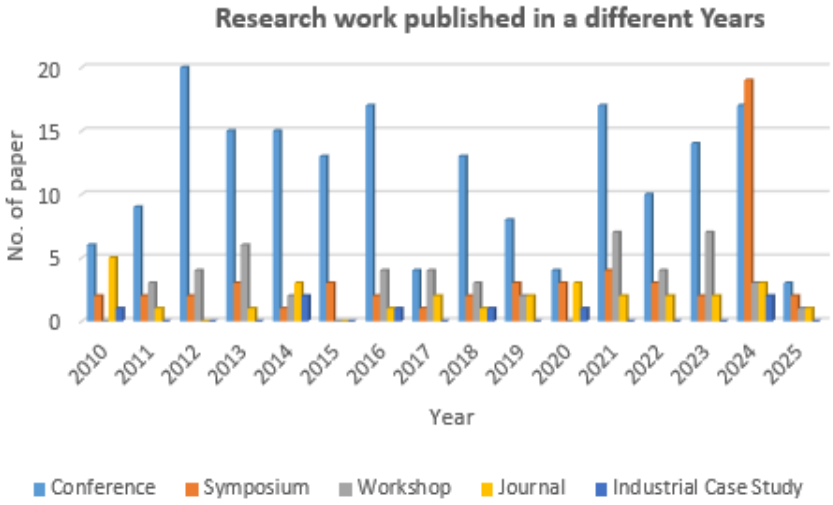

Figure 4. Research work on SPL in different years

CONCLUSION AND FUTURE WORK

The extensive variety of product variants arising from a product line renders SPL testing a demanding endeavor. Testing each individual product is impractical due to the sheer volume of products available. To mitigate the costs and time associated with software product testing, TCS techniques are employed. These techniques facilitate the scheduling of test cases for execution in a specific order, while also striving to enhance their efficiency in meeting performance objectives. In this paper, we present a survey focused on TCS techniques applicable to Software Product Lines (SPLs). This survey encompasses numerous research papers; however, we identified a limited number of studies directly related to our area of interest. There is a scarcity of literature addressing SPL-specific test case selection perspectives, and we have analyzed the contributions of all relevant papers. We propose that future research could be informed by our analysis and comparison. This survey reveals that the majority of research on SPLs has concentrated on addressing specific, narrow challenges. In order to increase the efficiency and accuracy of test selection, advanced machine learning models like as deep learning, reinforcement learning, and evolutionary algorithms will be integrated into test case selection methods for Software Product Lines (SPL) utilizing AI in the future. Additionally, there is a chance to create self-learning and adaptive systems that adjust to changes in the SPL and require less human involvement. Furthermore, hybrid approaches which combine AI with conventional methods might offer improved coverage and scalability.

However, there are a number of drawbacks to the methods used now. The high computational cost of training AI models, especially for extremely large and complex SPLs, is one of the main challenges. The absence of varied and high-quality training datasets is another drawback that may have an impact on model performance. Additionally, it is still difficult for testers to evaluate or trust the results due to the interpretability and openness of AI decisions in test selection. Last but not least, it is still challenging to generalize AI-based techniques across other SPL areas without requiring substantial modification.

REFERENCES

1. Ensan, A., et al. Goal-oriented test case selection and prioritization for product line feature models. In *Information Technology: New Generations (ITNG)*. 2011. IEEE. DOI:10.1109/ITNG.2011.58
2. Wang, S., et al. Automated test case selection using feature model: An industrial case study. In *ACM/IEEE 16th International Conference on Model Driven Engineering Languages and Systems (MODELS)*. 2013. IEEE. DOI:10.1007/978-3-642-41533-3_15
3. Beena, R., et al. Code coverage-based test case selection and prioritization. *International Journal of Software Engineering & Applications*, 2013. 4(6): p.29–40. DOI:10.5121/ijsea.2013.4604

4. Wang, S., et al. A systematic test case selection methodology for product lines: Results and insights from an industrial case study. *Journal of Software Engineering & Applications*, 2014. DOI:10.5121/ijsea.2013.4604
5. Wang, S., et al. Automated product line test case selection: Industrial case study and controlled experiment. *Software and Systems Modeling*, 2015. DOI:10.1007/s10270-015-0462-4
6. Ferreira, J.M., et al. Software product line testing based on feature model mutation. *International Journal of Software Engineering and Knowledge Engineering*, 2017. 27(5): p.817–839. https://doi.org/10.1142/S0218194017500309
7. Jammalamadaka, K., et al. Test case selection using logistic regression prediction model. *International Journal of Mechanical Engineering and Technology (IJMET)*, 2017. 8(11): p.786–796.
8. Markiegi, U., et al. Test case selection using structural coverage in software product lines for time-budget constrained scenarios. *Empirical Software Engineering*, 2019. DOI:10.1007/s10664-014-9345-5
9. Kumar, S., et al. Collaborative filtering-based test case prioritization and reduction for software product-line testing. In *IEEE Region 10 Conference (TENCON)*. 2019. Kochi, India: IEEE. DOI:10.1109/TENCON.2019.8929705
10. Jung, P., et al. Automated code-based test selection for software product line regression testing. *Journal of Systems and Software*, 2020. 158:110419. DOI:10.1016/j.jss.2019.110419
11. Jung, P., et al. Reducing redundant test executions in software product line testing—A case study. *Applied Sciences*, 2022. 10(23):8686. DOI:10.3390/app10238686
12. Raju, S.K., et al. Test case selection through novel methodologies for software application developments. *Symmetry*, 2023. 15(10):1959. DOI:10.3390/sym15101959
13. Khaleel, S., et al. Building a tool for optimal test cases selection using artificial intelligence techniques. 2023.
14. Padmanabhan, M. A systematic review of AI-based software test case optimization. *International Research Journal of Multidisciplinary Scope*, 2024. 5(4): p.847–859. DOI:10.47857/irjms.2024.v05i04.01451
15. Kumar, S., Kaushik, M., & Dubey, V. Test suite reduction techniques for software product line testing: A survey. *International Journal of Services, Economics and Management*, 2025. DOI:10.1504/IJSEM.2024.10067156
16. Islam, M., Khan, F., Alam, S., & Hasan, M. Artificial intelligence in software testing: A systematic review. In *IEEE Region 10 Conference (TENCON)*. 2023. DOI:10.1109/TENCON58879.2023.10322349
17. Ueda, K., Shikama, R., & Shimizu, Y. Automatic test cases generation with selection of training data for various system specifications. In *6th International Conference on Computer Communication and the Internet (ICCCI)*. 2024. IEEE. DOI:10.1109/ICCCI62159.2024.10674503
18. Kumar, S., & Kumar, R. Test case prioritization techniques for software product line: A survey. In *International Conference on Computing, Communication and Automation (ICCCA)*. 2016. IEEE. pp.884–889. DOI:10.1109/CCAA.2016.7813841
19. Kumar, S., & Kumar, R. Cost-based test case prioritization technique for software product line. *International Journal of Scientific Progress and Research*, 2017. 40(115): p.206–212.
20. Kumar, S., Kumar, R., & Mittal, M. A hybrid approach to perform test case prioritization and reduction for software product line testing. *International Journal of Vehicle Autonomous Systems*, 2020. 15(3–4): p.197–224. DOI:10.1504/IJVAS.2020.10039654
21. Kumar, S., Kumar, R., & Rani, M. Collaborative filtering-based test case prioritization and reduction for software product-line testing. In *IEEE Region 10 Conference (TENCON)*. 2019. pp.498–503. https://doi.org/10.1109/TENCON.2019.8929705
22. Kumar, S., Mittal, M., & Yadav, V.K. Cost-effective product prioritization technique for software product line testing. *International Journal of Engineering Systems Modelling and Simulation*, 2021. 12(2–3): p.83–93. DOI:10.1504/IJESMS.2021.115518
23. Saini, A., Kumar, R., & Kumar, S. A systematic literature survey on software product line testing. *International Journal of Research and Analytical Reviews (IJRAR)*, 2019. 6(1): p.253–262. https://doi.org/10.1729/Journal.19308